IN THE SPECIFICATION:

Please amend the specification as follows:

(1) The paragraph from page 2, line 6 to page 2, line 19 has been amended as follows:

The basic design of the event tester is disclosed in U.S. Patent Nos. 6,532,561 and ~~6,360,341~~ 6,360,343, which is briefly described here. An example of basic structure in the event based test system is shown in a block diagram of Figure 1. In the example of Figure 1, the event based test system includes a host computer 12 and a bus interface 13 both are connected to a system bus 14, an internal bus 15, an address control logic 18, a failure memory 17, an event memory 30 consisting of an event count memory (event count RAM) 20 and an event vernier memory (event vernier RAM) 21, an event summing and scaling logic 22, an event generator unit 24, and a pin electronics 26. The event based test system evaluates a semiconductor device under test (DUT) 28 connected to the pin electronics 26.

(2) The paragraph from page 5, line 5 to page 5, line 17 has been amended as follows:

In the U.S. Patent Nos. 6,360,343 and 6,557,133 and U.S. Application No. 10/318,959 (Publication No. US-2003-0229473), owned by the same assignee of this invention, it is disclosed an event summing and scaling logic for calculating a timing of the current event using the event data from the event memory.

In the event summing and scaling logic disclosed in the prior inventions, however, high speed reproduction of events was not fully established with use of pipeline processing. Further, compression technology is used for storing the event data in the event memory for saving the memory space. In the event summing and scaling logic disclosed in the prior inventions, high speed processing of decompressed vernier events is not fully established with use of parallel pipelines.

(3) The paragraph from page 6, line 29 to page 7, line 2 has been amended as follows:

The event count delay logic includes an event counter which loads the event count data and down-counts the event count data by the clock and produces a terminal count signal when a count result reaches a predetermined value, and an event count state machine which controls an overall operation of the event pipeline and summing logic including a process for loading the event count data into the event counter and a process for generating an event trigger signal in response to the terminal count signal from ~~he~~ the event counter.

(4) The paragraph from page 8, line 26 to page 8, line 29 has been amended as follows:

Figure 1 is a schematic block diagram showing a basic structure of an event based test system in the conventional technology for implementing the event pipeline and summing method and apparatus of the present invention.

(5) The paragraph from page 8, line 30 to page 8, line 32 has been amended as follows:

Figures 2A-2D are timing charts showing a basic relationship between a reference clock and timings of events in the event based test system in the conventional technology.

(6) The paragraph from page 9, line 1 to page 9, line 5 has been amended as follows:

Figure 4A is a timing charts showing an example of series of events, and Figure 4B is a schematic block diagram showing a concept of event data structure stored in an event memory which describe events for generating the series of events shown in Figure ~~4B~~ 4A.

(7) The paragraph from page 9, line 6 to page 9, line 9 has been amended as follows:

Figure 5 is a block diagram showing a basic structure of the event count delay unit and event vernier summing unit of Figure ~~3B~~ 3 for producing the waveform of Figure 4A with use of the event data of Figure 4B.

(8) The paragraph from page 9, line 33 to page 10, line 3 has been amended as follows:

Figures 12A-12B are block diagrams showing a basic architecture of an event vernier data decompression logic of Figure ~~3B~~ 3 wherein Figure 12A shows an event vernier state machine and Figure 12B shows a parallel circuit arrangement

for decompressing one or more pieces of vernier data for each event.

(9) The paragraph from page 10, line 8 to page 10, line 11 has been amended as follows:

Figure 14 is a block diagram showing an example of structure in the event vernier summing logic of Figure ~~3B~~ 3 for accumulating the vernier data of the series of events through the parallel pipelines.

(10) The paragraph from page 13, line 27 to page 14, line 2 has been amended as follows:

In the data table of Figure 4B, each of the events E0-E8 ~~(Figure 4A)~~ is defined by a time difference from the immediately prior event. Such time differences are denoted by ΔV_0, ΔV_1, ΔV_2, ...ΔV_8 in the waveform chart of Figure 4A. Since each time difference is a time length between two adjacent events, such a time difference is expressed by a combination of the event count data and the event vernier data. The event count data C0-C8 is shown in ~~a~~ an "Event Count" column and the vernier data V0-V8 is shown in ~~a~~ an "Event Vernier" column of the table. An "Event Type" column defines a type of event for each event such as "Drive Low" (1 to 0) and "Drive High (0 to 1).

(11) The paragraph from page 18, line 1 to page 18, line 10 has been amended as follows:

The event count state machine (ECS) 41 controls the operation of both the event counter 42 and an event vernier pipeline (ex. vernier data decompression 35 and event vernier summing 36 of Figure ~~3B~~ 3). The preferred embodiment of the present invention incorporates the compression method similar to the example shown in Figure 6C as will be described later. Thus, the event count state machine 41 controls the process of loading the event count data into the event counter 42. Specifically, the event count state machine 41 provides the following functions:

(12) The paragraph from page 24, line 6 to page 24, line 17 has been amended as follows:

As noted above, ~~since~~ there are times when the vernier data for each event are required to retrieve from two separate memory addresses. It is ~~no~~ not physically possible to retrieve data from two separate addresses simultaneously when using a single port RAM. Therefore, a pre-fetch must be performed that retrieves the contents of two addresses during initialization. Thus, when a multiport RAM is used as an event vernier memory, the pre-fetch queue registers may not be necessary. When looping, the loop registers are used to restore the state of the pre-fetch registers. Such a restoration process is a requirement for the proper operation of the decompression logic.

(13) The paragraph from page 27, line 4 to page 27, line 14 has been amended as follows:

Figure 14 shows the basic architecture of the event vernier ~~summation~~ summing logic 36. All clock inputs use the master clock unless otherwise specified. The event vernier ~~summation~~ summing logic 36 consists of a vernier 0 accumulation logic, a vernier carry-sum state machine, and a vernier pipeline sum logic. The vernier pipeline sum logic includes pipelines 0-3 where each pipeline is configured by a plurality (ex. Cells 1-5) of registers "Reg" series connected with one another. The vernier 0 accumulation logic includes an accumulator consisting of an ALU (arithmetic logic unit) 75 and a register 76 for accumulation of each vernier 0 delay through a multiplexer (MUX) 73 of each event. In this example, only the lower 6 bits of the accumulation are maintained.

(14) The paragraph from page 27, line 29 to page 28, line 4 has been amended as follows:

The final operation performed by the event vernier ~~summation~~ summing logic 36 is to add the accumulated vernier 0 delay ($\Delta V_{0\ sum}$) to each of the remaining vernier delays (ΔV_{n1}, ΔV_{n2} and ΔV_{n3}). The event vernier summing logic 36 includes four pipelines 0-3 each having an arithmetic logic unit (ALU) for adding the accumulated vernier 0 delay to the remaining vernier delays. This example shows that each pipeline has

five cells (Cell 1-5), i.e., series connected registers through which the data is sequentially shifted by one cell at each master clock. The above noted process produces the final vernier delay (vernier sum data) for each vernier pipeline during non-scaling operations.

(15) The paragraph from page 31, line 17 to page 31, line 29 has been amended as follows:

Figures 17 and 18 show the basic architecture of the event vernier scaling logic. The vernier data are shifted through the pipelines ~~1-4~~ 0-3 in the parallel fashion by the timing of the master clock. As shown in Figures 17 and 18, each of the pipeline 0-3 is configured by a plurality of series connected registers "Reg". Figure 17 shows the vernier data accumulator used for adding the vernier data an integer number of times. On the other hand, Figure 18 shows the final compare of the scale count from the scale counter logic 91 of Figure 16 and the MSB's of the accumulator in Figure 17. The pipeline enables used in the event count scaling logic are not used in the event vernier scaling logic which is simply a data flow-through. The output from previous portions of the event pipeline and summing logic 33 already enter this logic correctly delayed in time.

(16) The paragraph from page 32, line 16 to page 32, line 24 has been amended as follows:

Thus, in the event vernier ~~summing~~ scaling logic of Figure 17, the vernier sum data from the vernier summing logic of Figure 14 is supplied to the corresponding pipelines. The first cell (Cell 5) of each pipeline is an accumulator consisting of an arithmetic logic unit and a register. To multiply the vernier sum data by a scale factor "k", each accumulator repeats the accumulation cycles by "k-1" times as noted above. The scaled vernier sum data are shifted out from the pipelines (Cell 9).

(17) The paragraph from page 33, line 5 to page 33, line 19 has been amended as follows:

Figures 19A-19H illustrate these concepts for a scaling operation when the scale factor is "3". Figure ~~12A~~ 19A shows the master clock which is the primary clock to the event pipeline and summing logic 33 of the present invention. Figure ~~12B~~ 19B shows a scale count value which is an output signal of the scaling counter logic 91 in Figure 16 as the pipeline aligned count. In the preferred embodiment, since the scaling counter 91 is an up-counter, the scale count value increases at each master clock. Figure 19C shows an output of the vernier scale accumulator in Figure 17. As noted above, the vernier multiply operation is performed by the accumulator where the multiply operation simply consists of loading the vernier data followed by multiple addition. Thus, for the

scale factor "3", the vernier data V_n, for example, is added two times to create the scaled vernier data $3V_n$.

(18) The paragraph from page 35, line 23 to page 36, line 2 has been amended as follows:

Namely, the window strobe logic 38 generates a window strobe output when two event vernier data (vernier delay and event type) match. Figure 22 illustrates an example of basic architecture of the window strobe logic 38 in the preferred embodiment of the present invention. Figure ~~23~~ 25 illustrates an example of circuit diagram in the window strobe logic 38 for removing duplicate events. The window strobe logic 38 in Figures 22 and ~~23~~ 25 has a pipeline structure similar to the foregoing examples where event type data and vernier delay data are shifted through the parallel pipelines by the timing of the master clock. As shown in Figures 22 and 25, each pipeline includes a plurality of series connected registers "Reg".. This example shows the case where the window strobe is produced when both the vernier delay and event type of two events match with one another. When the window strobe is to be generated, the window strobe logic of Figure 22 produces a window strobe enable which is sent to the scaling logic of Figure 18 which performs the final output determination for the window strobe logic.

(19) The paragraph from page 36, line 17 to page 36, line 23 has been amended as follows:

The window strobe logic of Figure 22 receives event vernier data (vernier delay and event type signals) for each vernier pipeline directly from the event vernier state machine and event vernier pre-fetch queue shown in Figures 12A and 12B. Each data value is compared to all others. In the example of Figure 22, the comparison operation is conducted to ~~determined~~ determine the following:

(20) The paragraph from page 38, line 31 to page 39, line 8 has been amended as follows:

Thus, such duplicate events are removed by the circuitry shown in Figure 25. In this example, the removal operation is performed according to the rule listed in the table of Figure 26. The rule for this operation is that one of the two events must be removed. This table chooses the higher numbered Type and vernier to be eliminated. What this determination generates are a series of event enables. The event types and verniers will have the corresponding event enable signals disabled. A logical "AND" of the enable signals with the event valid signals determines which event data values will be invalidated (AND gates in Figure ~~26~~ 25). This effectively removes them from generating any operations in ~~th~~ the event generator.

(21) The paragraph from page 40, line 10 to page 40, line 13 has been amended as follows:

"ECR_CNT_LD[3:0]": event counter load strobes (event count state machine 41 and event counter 42 in Figure 7). The event counter 42 has four loadable segments (single word to quad word in Figure 9).